BAB 2
LANDASAN TEORI

2.1 Teori Umum

Teori umum merupakan suatu kumpulan teori-teori yang akan digunakan sebagai landasan teori atau pengertian dasar dari suatu konsep yang digunakan dalam suatu penelitian. Teori ini pada umum nya sebagai pelengkap dalam pengerjaan skripsi yang dibuat.

2.1.1 Sistem Informasi

Sistem adalah kumpulan dari sebuah komponen yang saling berhubungan dan mempunyai fungsi yang sama untuk mencapai beberapa hasil. (Satzinger, 2012, p7-p8). Semua sistem tersebut memiliki 3 fungsi dasar yaitu:

a. *Input* : melibatkan merekam dan mengumpulkan elemen yang masuk ke sistem untuk diproses. Misalnya, bahan mentah, energi, dan data.
b. Proses : melibatkan proses transformasi yang mengubah *input* menjadi *output*. Contohnya adalah proses manufaktur, atau perhitungan matematis.
c. *Output* : melibatkan proses mentransfer elemen yang telah dihasilkan oleh proses transformasi ke tujuan. Misalnya sebuah produk, informasi dan lain lain.

Sistem sendiri berasal dari bahasa Yunani (sustēma) atau (systēma). Ketiga fungsi dasar sistem tersebut tentunya sudah sering kita temukan pada kehidupan sehari – hari dan bukan hal sulit lagi untuk dipahami.

Berdasarkan pernyataan di atas, penulis menyimpulkan bahwa sistem merupakan komponen-komponen yang saling berkait satu sama lainnya yang berfungsi bersama-sama untuk mencapai hasil yang di

inginkan dimana terdapat *input* yang akan diproses untuk menghasilkan *output* yang dibentuk.

Informasi adalah data yang telah dikumpulkan, disimpan, dan diproses untuk menghasilkan sesuatu yang bermanfaat bagi orang lain (Satzinger 2012, p7). Berdasarkan definisi yang dijabarkan atau dibahas diatas oleh para ahli, maka dapat disimpulkan bahwa informasi adalah suatu data yang telah diproses menjadi bentuk yang lain dan berguna bagi pihak tertentu dalam mengambil sebuah keputusan yang telah diperoleh.

Sistem informasi adalah sekumpulan komponen yang saling berhubungan yang mengumpulkan, memproses, menyimpan, dan menyediakan *output* berupa informasi yang diperlukan untuk menyelesaikan tugas bisnis (Satzinger, 2012, p7-p8). Dari definisi diatas, dapat disimpulkan bahwa sistem informasi merupakan suatu kumpulan komponen atau unsur yang bersatu bersama-sama untuk menghasilkan dan mengelola informasi yang berguna untuk menyelesaikan tugas bisnis dan mendukung suatu organisasi dalam mengumpulkan informasi yang dibutuhkan.

2.1.2 Komponen Sistem Informasi

Menurut James O'Brien (2010, p32) yang menunjukkan kerangka konsep dasar untuk berbagai komponen dan aktivitas sistem informasi. Sistem informasi bergantung pada sumber daya manusia (pemakai akhir dan pakar sistem informasi), *hardware* (mesin dan media), *Software* (program dan prosedur), data (dasar data dan pengetahuan), serta jaringan (media komunikasi dan dukungan jaringan) untuk melakukan suatu *input*, pemprosesan, *output*, penyimpanan, dan aktivitas pengendalian yang mengubah sumber daya data menjadi produk informasi yang dibuat.

2.1.3 Teori Pengumpulan Data

Perancangan sebuah sistem harus berdasarkan data hasil penelitian tentang masalah yang pengguna alami, oleh karena itu diperlukan proses pengumpulan data dengan teknik yang tepat. Menurut (Satzinger, 2012, p48) ada beberapa teknik pengumpulan data yaitu :

a. Melakukan wawancara dengan *stakeholders* dan pengguna
   Teknik wawancara *stakeholders* dan pengguna ini sangat efektif untuk menemukan kunci permasalah sekaligus memprediksikan bagaiman ekspekasi calon pengguna nanti saat sistem sudah bisa digunakan. Proses wawancara juga membantu kami untuk mengetahui proses bisnisnya.
b. Mendistribusikan dan menyebar kuisioner
   Mengirim kuisioner kepada para calon pengguna merupaka cara yang efektif untuk mengumpulkan banyak data dari berbagai tempat jika perusahaan tersebut terletak dibanyak lokasi yang terpisah jauh secara geograpis.
c. Meninjau *input, output* dan proses
   Meninjau jurnal-jurnal, dokumen bisnis atau majalah perusahaan juga bisa menjadi sumber informasi yang bermanfaat sebagai referensi saat memberikan solusi pada pengguna. Setelah meninjau dokumen-dokumen tersebut kita bisa mengamati proses bisnis nya apakah sudah efektif atau sebenarnya bisa lebih efisien lagi.
d. Obeservasi dan mendokumentasikan proses bisnis
   Obeservasi dokumen bisnis adalah sumber informasi yang sangat berharga karena dalam dokumen tersebut kita bisa mendapatkan informasi bagaimana sebenarnya perusahaan tersebut menjalankan bisnis*nya* dan apa yang sebenarnya terjadi saat ini.
e. Melakukan penelitian dan memilih perusahaan lain menyelesaikan permasalahan.

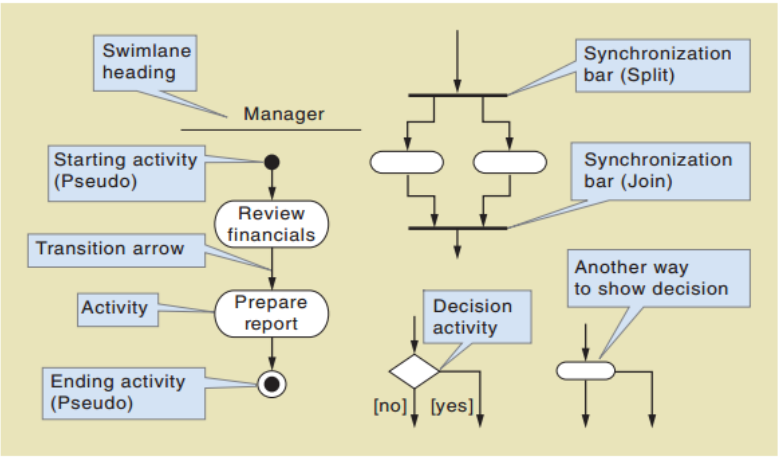
Kita bisa juga meminta bantuan vendor untuk menyelesaikan masalah disebuah perusahaan karena vendor tersebut sudah pernah mengatasi masalah yang sama dan sukses. Jika cara ini lebih efisien daripada melakukan penelitian dari internal, maka cara ini sebaiknya digunakan.

f. Mengumpulkan saran dan kritik dari para pengguna
   Saat sistem sudah hampir jadi maka akan ada proses mencoba. Setelah para pengguna mencoba menggunakan sistem yang baru maka akan muncul saran dan kritik agar sistem menjadi lebih baik lagi.

2.1.4 *Unified Modeling Language*

UML adalah model standar dari perancangan dan notasinya yang melakukan pengembangan orientasi secara spesifik yang ditentukan oleh *Object Management Group* (OMG) organisasi standar untuk pengembangan sistem. Dengan menggunakan UML, analis, dan akhirnya pengguna dapat menggambarkan dan memahami berbagai diagram khusus yang digunakan dalam proyek pengembangan sistem, model-model komponen sistem yang berbasiskan *Unified Modeling Language* terdiri dari lima diagram yaitu *use case diagram, sequence diagram, communication diagram, dan state machine diagram* (Satzinger, 2012, p46)

2.1.4.1 *Activity Diagram*



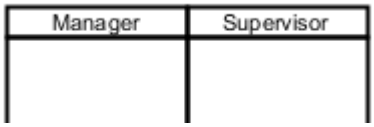
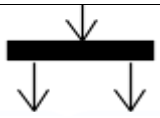
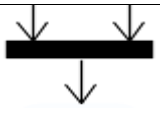
Gambar 2. 1 Simbol *Activity Diagram*

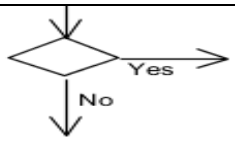
(Satzinger et. Al,2012 : 58)

Activity Diagram menggambarkan aktivitas dari beberapa *user* (atau sistem), orang yang melakukan tiap-tiap aktivitas dan urutan aliran dari aktivitas tersebut pada sebuah sistem. (Satzinger, 2012, p57)

Berikut simbol-simbol yang digunakan pada *activity diagram:*

Tabel 2. 1 Simbol-simbol *activity diagram*

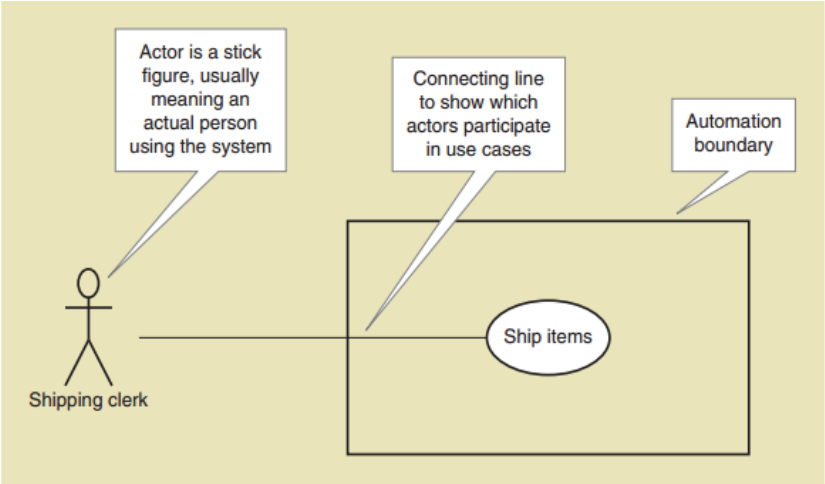
Notasi	Deskripsi
 *Swimlane heading*	Menunjukan aktor atau sistem yang melakukan aktivitas.
 *Start node*	Notasi memulai *activity diagram*.
 *Activity*	Gambaran sebuah aktivitas.
 *Control flow*	Menunjukan alur dari aktivitas.
 *Split*	Sebuah aktivitas dapat dipecah menjadi beberapa aktivitas yang berjalan secara parallel.
	Beberapa aktivitas akan bergabung untuk melaksanakan aktivitas

Notasi	Deskripsi
Join	selajutnya.
 *Decision*	Sebuah aktivitas dapat mempunyai alternatif pilihan.
 *End node*	Notasi mengakhiri *activity diagram*.

(Sumber: Satzinger, 2012)

2.1.4.2 *Use case Diagram*

Use case diagram adalah model UML yang digunakan untuk menggambarkan hubungan antara pengguna dan *use case* sistem (Satzinger, 2012, p78)



Gambar 2. 2 Contoh *Use case Diagram* sederhana dengan actor

(Sumber: Satzinger et. al (2012 : 81)

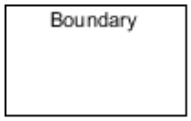
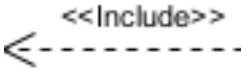
Komponen pembentuk *use case diagram* adalah, sebagai berikut:

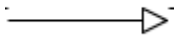
1. Aktor (*An actor*), menggambarkan pihak-pihak yang berperan dalam sistem.

2. *A Use case*, aktifitas/sarana yang disiapkan oleh bisnis/sistem.
3. *A Sistem boundary*, sebuah kotak yang mewakili sebuah sistem.
4. Hubungan (*An association relationship*), aktor mana saja yang terlibat dalam *use case*, dan bagaimana hubungan *use case* dengan *use case* lain. Ada hubungan antar *use case*. Digolongkan menjadi dua: yaitu *extend* digambarkan dengan keterangan <<*extend*>>, dan *include* digambarkan dengan keterangan <<*include*>>.

Berikut simbol-simbol yang digunakan pada *use case diagram:*

Tabel 2. 2 Simbol-simbol *use case diagram*

Notasi	Deskripsi
 *Actor*	Orang yang menggunakan sistem.
 *Boundary*	Ruang lingkup otomasi.
 *Usecase*	Kegiatan atau aktifitas yang dapat dilakukan *actor.*
Association relationship	Menghubungkan *actor* dengan *use case.*
 *Include*	Menunjukan sebuah *use case* memerlukan *use case* lain untuk menjalankan fungsinya.

Notasi	Deskripsi
 *Extend*	Menunjukan sebuah *use case* dapat berdiri sendiri walaupun tanpa *use case* tambahan.
 *Generalization*	Hubungan *child use case* ke *parent use case*. Menentukan *child use case* mendapat turunan perilaku dan karakteristik dari *parent use case*

(Sumber: Satzinger, 2012)

2.1.4.3 *Fully Use case Description Diagram*

Use case Description, menurut Satzinger (2010, p121), adalah sebuah model tekstual yang merinci dan mendeskripsikan proses detail dari sebuah *use case*. Berikut *form* yang digunakan pada *Fully Use case Description diagram*

Tabel 2. 3 *Fully Use case Description table*

Use case name:		
Scenario:		
Triggering event:		
Brief Description:		
Actors:		
Related *use cases*:		
Stakeholders:		
Preconditions:		
Postconditions:		
Flow of activities:	**Actor**	**System**
Exception conditions:		

(Sumber: Satzinger, 2012)

Kolom pertama dan kedua digunakan untuk mengidentifikasikan *use case* dan skenario dalam *use case* yang sedang didokumentasikan. Kolom ketiga mengidentifikasikan penyebab terjadinya *use case* tersebut. Kolom keempat adalah penjelasan singkat atas *use case* atau skenario yang terjadi. Kolom kelima untuk mengidentifikasikan aktor yang terlibat. Kolom keenam untuk mengidentifikasikan *use case* lain yang berhubungan atau terlibat dalam *use case* yang sedang didokumentasikan. Kolom ketujuh mengidentifikasikan stakeholders yang tidak harus terlibat dalam *use case* tapi tertarik pada hasil dari *use case* tersebut. Kolom kedelapan dan kesembilan adalah *Preconditions* dan *Postconditions*. *Preconditions* mengidentifikasikan kondisi apa yang harus ada untuk *use case* dapat berjalan. *Postconditions* mengidentifikasi apa yang akan terjadi setelah penyelesaian *use case*. Kolom kesepuluh mengidentifikasikan alur lengkap dari aktifitas *use case*. Aktifitas *alternative* dan kondisi di identifikasikan ke dalam kolom terakhir.

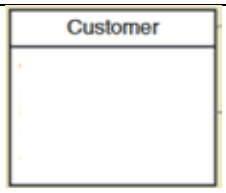
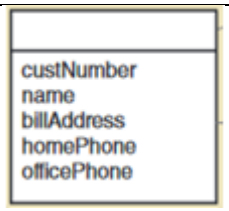
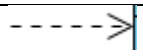
Use case name:	*Ship items*.	
Scenario:	Ship items for a new sale.	
Triggering event:	Shipping is notified of a new sale to be shipped.	
Brief description:	Shipping retrieves sale details, finds each item and records it is shipped, records which items are not available, and sends shipment.	
Actors:	Shipping clerk.	
Related use cases	None.	
Stakeholders:	Sales, Marketing, Shipping, warehouse manager.	
Preconditions:	Customer and address must exist. Sale must exist. Sale items must exist.	
Postconditions:	Shipment is created and associated with shipper. Shipped sale items are updated as shipped and associated with the shipment. Unshipped items are marked as on back order. Shipping label is verified and produced.	
Flow of activities:	**Actor**	**System**
	1. Shipping requests sale and sale item information.	1.1 System looks up sale and returns customer, address, sale, and sales item information.
	2. Shipping assigns shipper.	2.1 System creates shipment and associates it with the shipper.
	3. For each available item, shipping records item is shipped.	3.1 System updates sale item as shipped and associates it with shipment.
	4. For each unavailable item, shipping records back order.	4.1 System updates sale item as on back order.
	5. Shipping requests shipping label supplying package size and weight.	5.1 System produces shipping label for shipment. 5.2 System records shipment cost.
Exception conditions:	2.1 Shipper is not available to that location, so select another. 3.1 If order item is damaged, get new item and updated item quantity. 3.1 If item bar code isn't scanning, shipping must enter bar code manually. 5.1 If printing label isn't printing correctly, the label must be addressed manually.	

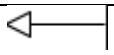
Gambar 2. 3 Contoh *Fully Use case Description* (Satzinger, 2012)

2.1.4.4 *Domain Model Class Diagram*

Class merupakan kategori atau klasifikasi dari kumpulan objek atau benda. Sedangkan *domain class* merupakan *class* yang mendeskripsikan objek dari problem *domain*. Pada UML *class diagram* digunakan untuk menunjukan *class* dari objek-objek pada sistem, sehingga *domain model class diagram* adalah *class diagram* yang menunjukan problem *domain* dari pengguna (Satzinger, 2012). Berikut simbol-simbol yang digunakan pada *domain model class diagram*:

Tabel 2. 4 Simbol-simbol *domain model class diagram*

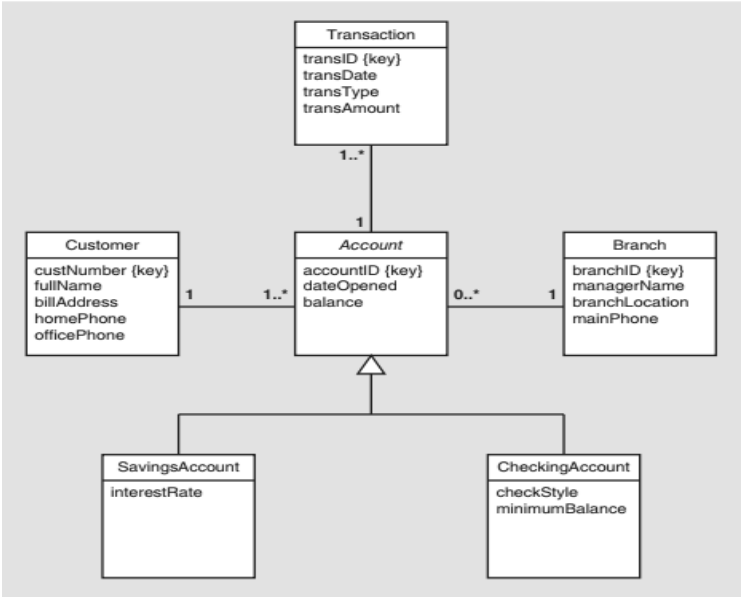
Notasi	Deskripsi
 *Class*	Nama dari *class*
 *Attributes*	Nilai atau atribut dari suatu objek dalam *class*
 *Asosiasi*	Hubungan statis antara dua *class* bersama dengan multiplisitas
 *Dependency*	Relasi dimana sebuah *class* membutuhkan *class* lainnya untuk dapat berjalan dalam hal ini dapat berbentuk parameter *object* yang dieksekusi dalam

Notasi	Deskripsi
	method class lainnya
 *Generalization*	Fitur warisan dari konsep berorientasi objek. Dimana *child* mewarisi atribut dan *method* dari *parent*nya
 *Composition*	Suatu *class* merupakan bagian utuh dari *class* lainya namun pada hal ini satu bagian *class* tersebut akan sangat bergantung pada keberadaan *class* lainya
 *Aggregation*	Relasi dimana sebuah *class* merupakan bagian utuh dari *class* lainya sering digambarkan dengan kata "*has a*" berarti memiliki
0..1 *Zero or one*	*Optional*, nol atau satu objek
0..* *Zero or more*	*Optional*, nol atau lebih banyak objek
1 *One and one only*	*Mandatory*, tepat satu objek
* *Zero or more alternate*	*Optional*, banyak objek
1..1 *One and one only alternate*	*Mandatory*, tepat satu objek
1..* *One or more*	*Mandatory*, satu atau lebih banyak objek

Berikut langkah-langkah membuat *domain model class diagram* yaitu:

1. Mengidentifikasi semua *class* yang muncul.
2. Menentukan atribut dari setiap *class*.
3. Membuat dan menentukan hubungan dan tipe relasi antar *class*.
4. Menambahkan multisiplitas pada relasi *class*.

Berikut contoh *domain model class diagram* pada sistem perbankan yaitu:



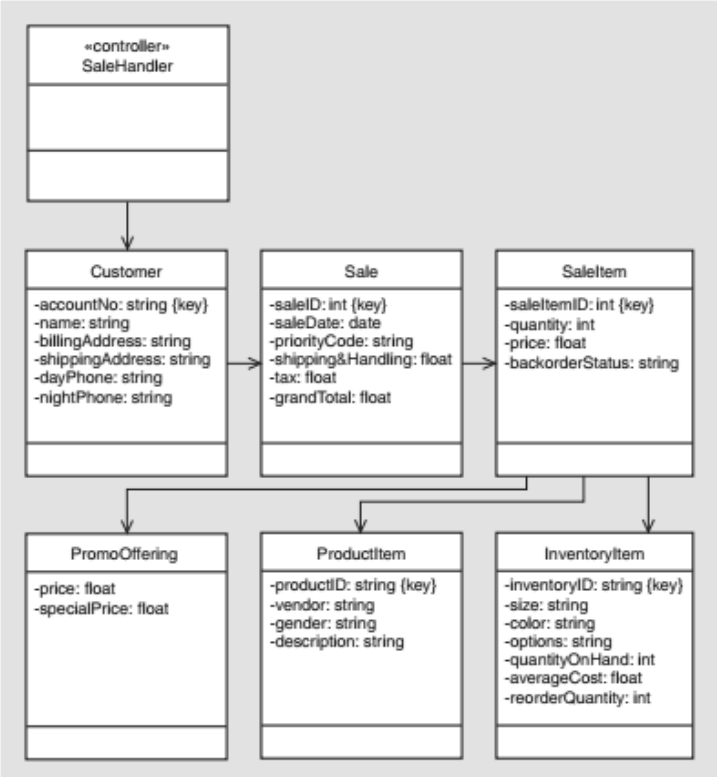
Gambar 2. 4 Contoh *domain model class diagram* (Satzinger, 2012)

a. First-Cut Design Class Diagram

First Cut Design Class Diagram merupakan perluasan dari *model domain class diagram* (Satzinger, 2012). Perluasan tersebut diperlukan dua tahap, diantaranya yaitu:

1. Mengelaborasi atribut-atribut dengan tipe dan nilai informasi inisial.
2. Menambahkan panah navigasi visibilitas.

Berikut contoh *first cut design class diagram* pada sistem penjualan telepon yaitu:



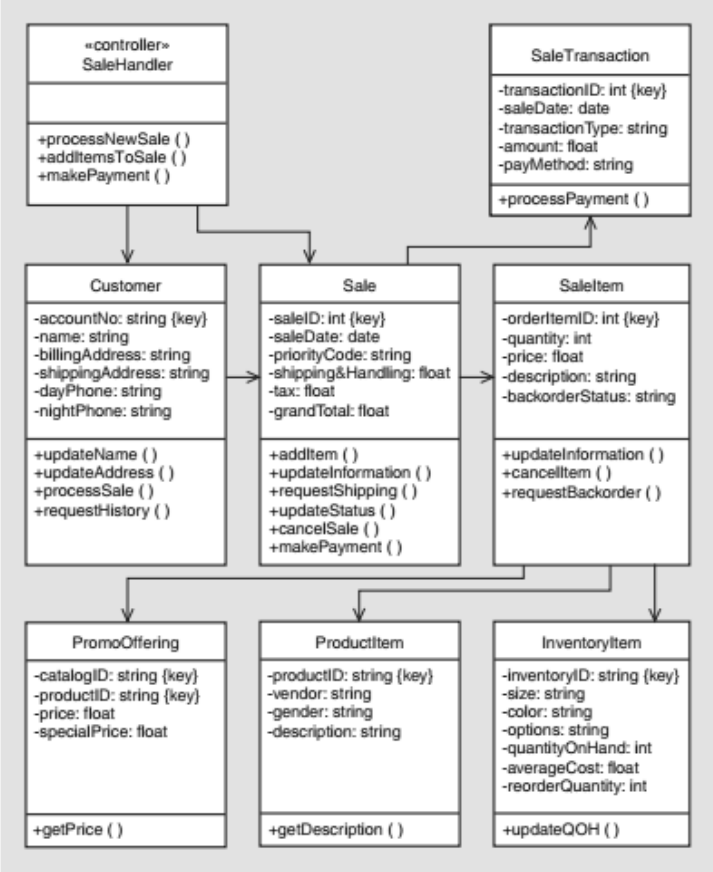
Gambar 2. 5 Contoh *First Cut Design Class Diagram* (Satzinger, 2012)

b. Update Design Class Diagram

Updated design class diagram adalah sebuah *class diagram* lanjutan dari *first-cut design class diagram* yang lebih *detail* dalam menjelaskan *input message* yang terdapat pada *first cut sequence diagram*, alur data beserta tipe datanya, dan *input message* yang akan dilaksanakan oleh *use case controller* (Satzinger, 2012).

1. Membuat *controller* yang akan menjadi *method* untuk merubah data *class*.
2. Menuliskan detail atribut *class controller* beserta data tipenya.
3. Menghubungkan *class* tersebut dengan arah panah sesuai *first cut sequence diagram.*

Berikut contoh *updated design class diagram* pada sistem penjualan telepon yaitu:



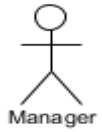
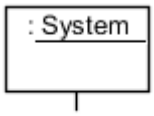
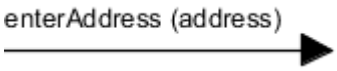
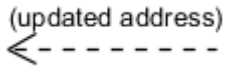
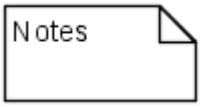
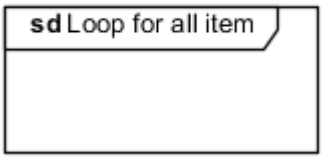
Gambar 2. 6 Contoh *Updated design class diagram* (Satzinger, 2012)

2.1.4.5 *System Sequence Diagram*

System sequence diagram digunakan untuk menggambarkan hubungan aktor dan sistem yang menyajikan *flow* dari informasi *input* dan *output* dari sebuah sistem otomasi (Satzinger, 2012).

Berikut simbol-simbol yang digunakan pada *system sequence diagram*:

Tabel 2. 5 Simbol-simbol *system sequence diagram*

Notasi	Deskripsi
 *Actor*	Aktor eksternal yang berinteraksi dengan sistem.
 *System object*	Objek yang mewakili sistem otomasi.
Lifeline	Menunjukan alur dari *message* dari atas ke bawah.
 *Input message*	Message *input* dari aktor.
 *Output message*	Hasil *output* atau *return value* dari sistem.
 *Optional note*	Keterangan tambahan untuk menjelaskan sesuatu pada diagram.
 *Repeatability*	Pengulangan untuk suatu kondisi dalam kotak.

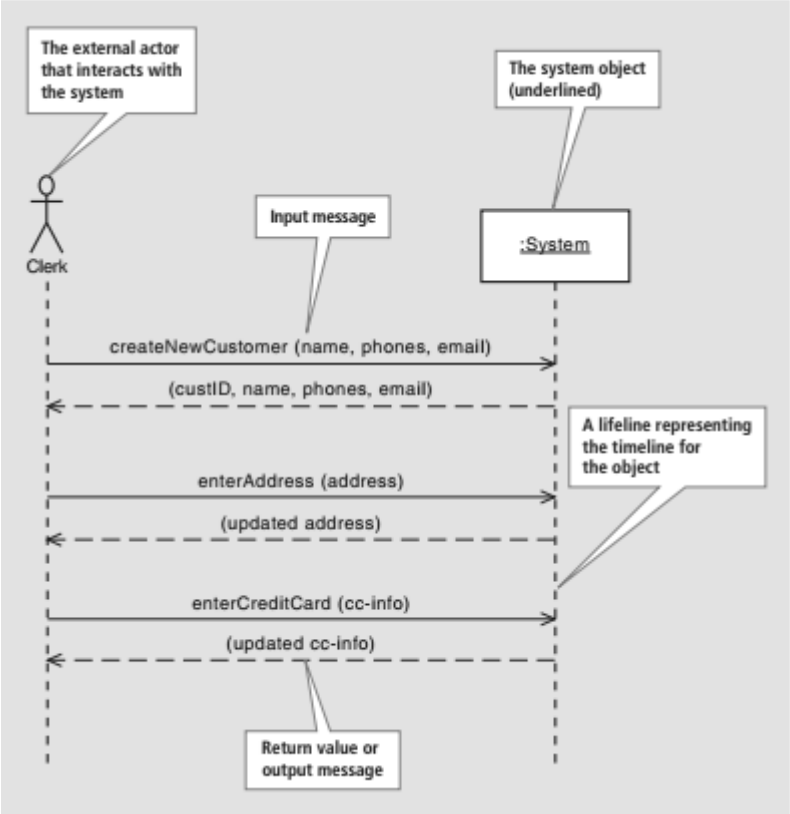
(Sumber: Satzinger, 2012)

Berikut langkah-langkah membuat *system sequence diagram* yaitu:

1. Aktor dan alur yang digunakan berdasarkan *activity diagram*.
2. Mengidentifikasi pesan *input*, dapat dilihat pada arah panah *activity diagram*.
3. Mendeskripsikan pesan dari aktor eksternal ke sistem menggunakan pesan notasi di atas.

4. Mengidentifikasi dan menambah kondisi *input* pesan, termasuk iterasi dan kondisi benar atau salah.
5. Setelah *input* diketahui, identifikasi *output* pesan kembalian dari sistem.

Berikut contoh dari *system sequence diagram* pada proses pembuatan akun pelanggan yaitu:



Gambar 2. 7 Contoh ***system sequence diagram*** **(Sumber: Satzinger, 2012)**

Multi layer digunakan untuk mendukung jaringan *multitier* yang mana *database server* berada dalam satu mesin, dan *busines logic* nya ada pada *server* lain, dan *user interface* ada di mesin *desktop* (Satzinger, 2012). Adapun tiga lapisan tersebut adalah sebagai berikut:

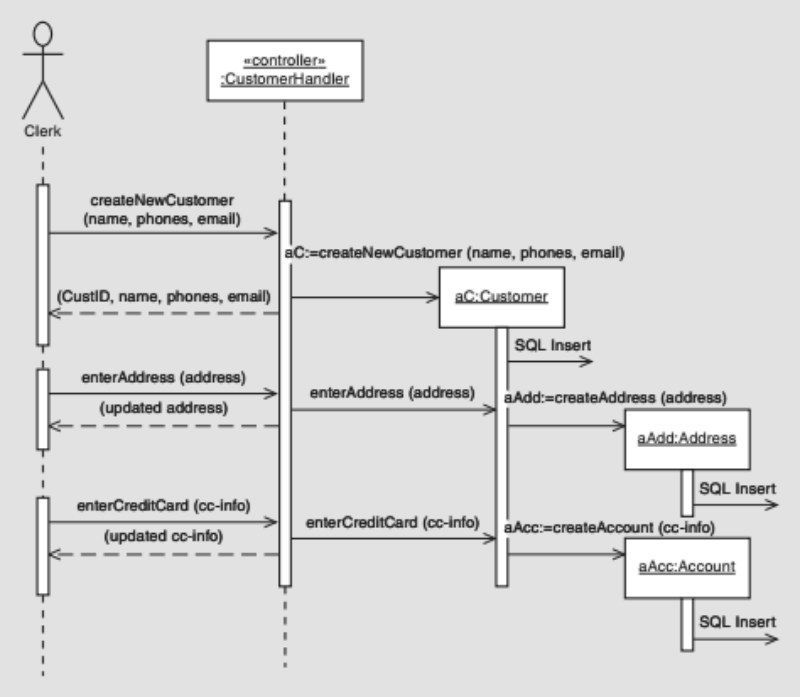
a. First Cut Sequence Diagram

First cut diagram merupakan *sequence digram* yang *detail*, menggunakan semua elemen pada sy*stem sequence diagram*. Perbedaannya objek sistem diganti dengan objek internal dan pesan antar sistem (Satzinger, 2012).

Berikut langkah-langkah tambahan dalam membuat f*irst cut sequence diagram* yaitu:

1. Mengambil semua pesan *input* dan menentukan pesan internal yang dihasilkan dari *input*
2. Mengidentifikasi kumpulan *class* yang dipengaruhi oleh pesan.
3. Menyempurnakan komponen tiap pesan, menambahkan iterasi, kondisi benar atau salah, nilai kembalian, dan parameter.

Berikut contoh dari f*irst cut sequence diagram* pada proses penambahan akun pelanggan oleh pegawai yaitu:



Gambar 2. 8 Contoh ***first cut sequence diagram*** **(Sumber: Satzinger, 2012)**

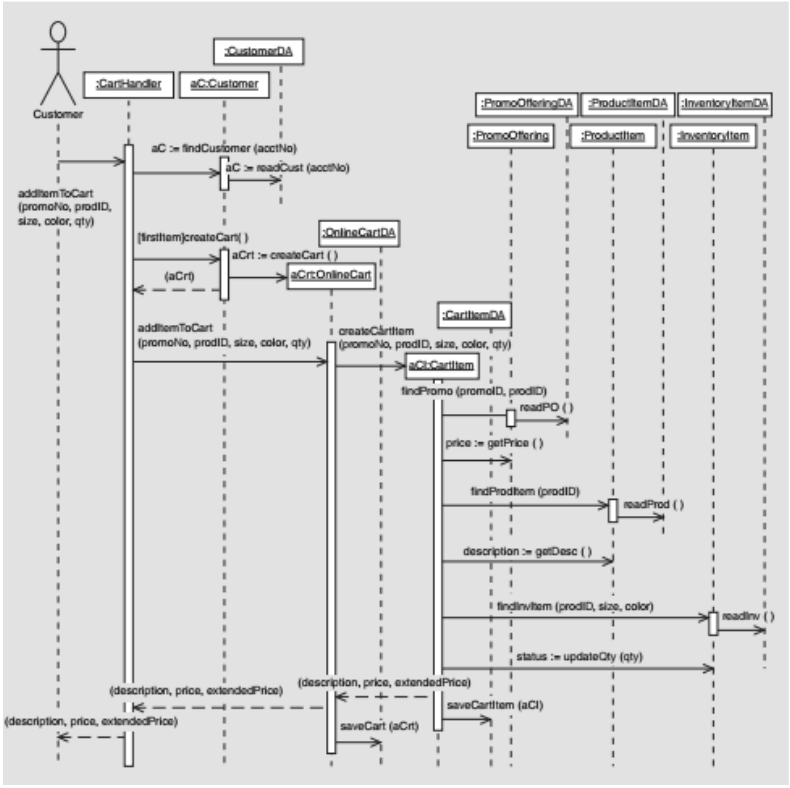
b. Data Access Layer

Data access layer diperlukan pada *business logic* yang kompleks dan harus diisolasi dari SQL *statement* yang mengakses ke *database* (Satzinger, 2012).

Berikut langkah-langkah tambahan dalam membuat *data access layer sequence diagram* yaitu:

1. Menambahkan *constructor method* setiap *object problem domain*.
2. Menambahkan pengiriman *message* ke objek *data access layer* dan membaca ke *database* intansiasi *problem domain object*.

Berikut contoh dari *data access layer sequence diagram* pada proses pembelian oleh pelanggan yaitu:



Gambar 2. 9 Contoh ***data access layer sequence diagram*** **(Sumber: Satzinger, 2012)**

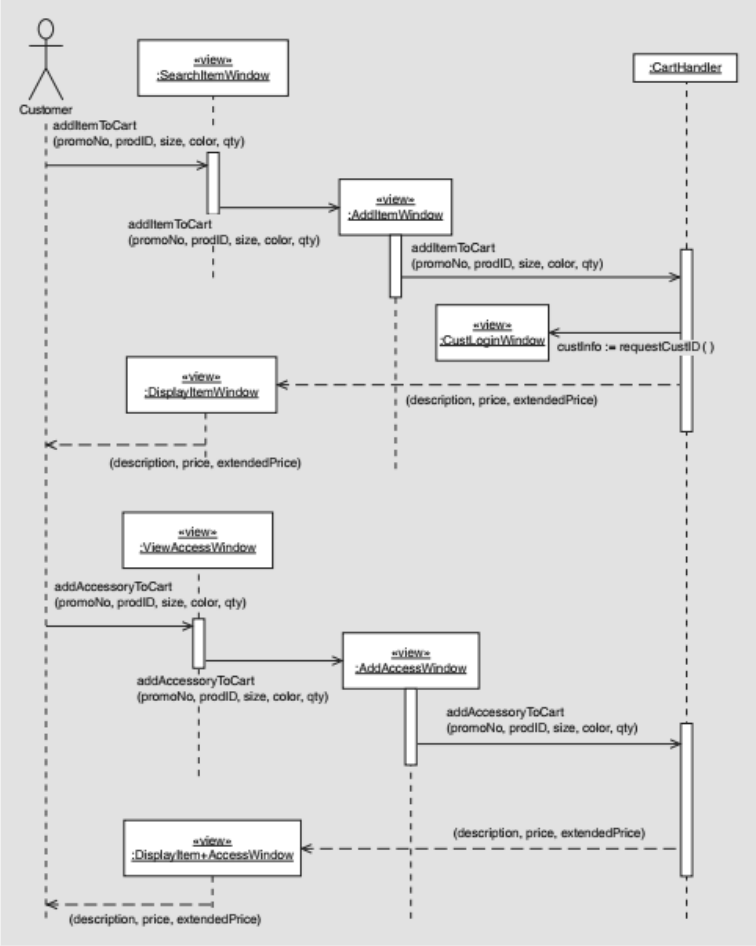
c. View Layer

View layer melibatkan interaksi antara pengguna dan komputer dengan *user interface* pada masing-masing *use case* nya.

Berikut langkah-langkah tambahan dalam membuat *view layer sequence diagram* yaitu dengan menambahan komponen *user interface*. Ada dua sumber *input* untuk desain *view layer* yaitu (Satzinger, 2012):

1. Desain *user interface.*
2. *First cut sequence diagram* atau *sequence diagram* dengan *data access* yang sudah diidentifikasi.

Berikut contoh dari *view layer sequence diagram* pada proses pembelian oleh pelanggan yaitu:



Gambar 2. 10 Contoh ***view layer sequence diagram*** **(Sumber: Satzinger, 2012)**

2.2 Teori Khusus

2.2.1 *System Development Life Cycle (SDLC)*

Menurut Satzinger (2012, p38) *System Development Life Cycle* merupakan seluruh proses yang membangun, menyebarkan, menggunakan dan memperbarui sistem informasi yang telah dibuat secara rinci. *System Development Life Cycle (SDLC)* mengidentifikasi semua aktivitas yang dibutuhkan untuk membangun, menjalankan,serta memelihara sebuah sistem informasi. Dalam Pengembangan SDLC, terdapat beberapa kelompok tahapan masing-masing dimana kegiatan

tersebut terdapat perbedaan. Tahapan-tahapan tersebut diantaranya adalah *Project Planning phase, Analysis phase, Design phase, Implementation phase,* dan *Support phase* (Satzinger, 2012, p40).

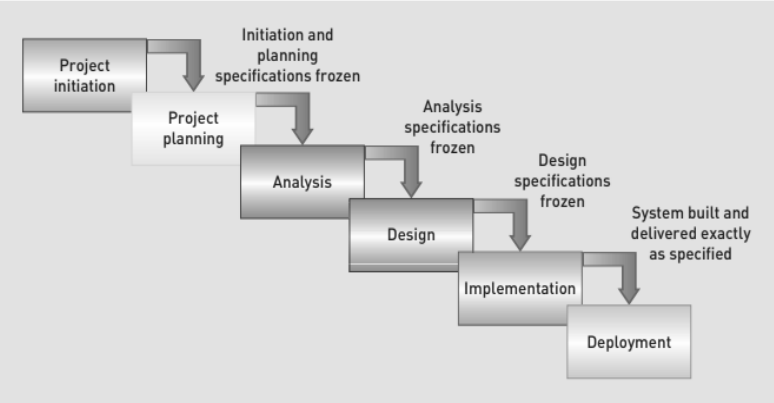
2.2.2 *Waterfall Methodology*

Waterfall Model adalah sebuah metode pengembangan *Software* yang bersifat sekuensial. Metode ini dikenalkan oleh Royce pada tahun 1970 dan pada saat itu disebut sebagai siklus klasik dan sekarang ini lebih dikenal dengan sekuensial linier. Selain itu model ini merupakan model yang paling banyak dipakai oleh para pengembang *Software*. Saat ini pendekatan Waterfall merupakan metodologi pengembangan perangkat lunak yang dominan di banyak perusahaan besar. Dalam pendekatan Waterfall, tanggung jawab sering ditempatkan dalam *Information System Function* (ISF) (Mahadevan, Lakshman : 2015).

Metode *waterfall*, dengan fase proyek mengalir turun, satu demi satu. Model ini mengasumsikan bahwa fase dapat dilakukan dan diselesaikan secara berurutan. Pertama, rencana terperinci dikembangkan, kemudian persyaratan ditentukan secara menyeluruh, kemudian sistem dirancang hingga algoritma terakhir, dan kemudian diprogram, diuji, dan dipasang.

Inti dari metode *waterfall* adalah pengerjaan dari suatu sistem dilakukan secara berurutan atau secara linear. Dalam model ini mengasumsikan bahwa langkah yang dilakukan dan diselesaikan secara berurutan, jadi jika langkah satu belum dikerjakan maka tidak akan bisa melanjutkan kelangkah 2, 3, dan seterusnya. Secara otomatis tahapan ke-3 akan bisa dilakukan jika tahap ke-1 dan ke-2 sudah dilakukan.

Menurut John Satzinger (Satzinger, 2012), Metodologi *waterfall* dapat di gambarkan seperti gambar berikut.



Gambar 2. 11 Metodologi waterfall (Satzinger, 2012)

1. *Initiation*

Tujuan dari tahap *initiation* adalah untuk melakukan investigasi tingkat awal dari kebutuhan bisnis dan menghasilkan rekomendasi sebagai solusinya. Setelah disetujui oleh tim manajemen, *stakeholders*, klien atau sponsor proyek, baru akan berlanjut ke tahap berikutnya.

2. *Planning* (Perencanaan)

Tahap *planning* adalah dasar pemahaman mengapa sistem informasi harus dibangun dan menentukan bagaimana tim proyek akan membangun proyek tersebut. Dalam tahapan *planning* terdiri dari dua tahap yakni *project initiation* dan *project approval*.

3. *Analysis* (Analisis)

Tahap analisis adalah proses untuk menjawab pertanyaan-pertanyaan mengenai siapa yang akan menggunakan sistem, apa yang akan sistem lakukan, dimana sistem digunakan, dan kapan sistem akan digunakan. Pada proses ini tim akan megidentifikasi proses bisnis yang saat ini berjalan, kekurangan, dan sistem informasi yang dibangun untuk kedepannya. Hasil dari tahap analysis adalah penjelasan *requirement definition* yang harus dipenuhi serta model dari sistem informasi yang baru. Hasil dari

tahap ini dapat digambarkan melalui *functional modelling, structural modelling*, dan *behavioral modelling*. *Functional modelling* untuk menggambarkan fungsi yang akan dibangun, *Structural modelling* menggambarkan struktur data yang mendukung proses bisnis. *Behavioral modelling* menggambarkan perubahan aspek internal.

4. *Design* (Desain)

Tahap desain adalah proses untuk memutuskan bagaimana sistem akan beroperasi dalam hal perangkat keras, perangkat lunak dan infrastruktur jaringan, antarmuka pengguna, dan *file* yang diperlukan. Hasil dari tahap ini adalah spesifikasi *Software* dan *hardware* yang dibutuhkan dan desain tampilan program.

5. *Implementation* (Implementasi)

Implementasi merupakan tahap akhir pada SDLC, di mana sistem adalah benar-benar dibangun. Pada tahap ini dikerjakan *coding program, testing program,* migrasi data, pelatihan, serta melakukan *maintenance* jika ada kerusakan.

6. *Deployment* (Pengembangan)

Pengembangan merupakan tahanp yang sifatnya terus menerus, dimana setelah kita berhasil mengimplementasikan rancangan sebuah program kita pasti akan terus melakukan analisa, evaluasi dan inovasi terhadap aplikasi atau produk yang sudah jadi tersebut. Setelah terjadi satu kali pengembangan maka ulangi lagi dilakukan analisa dan evaluasi yang menghasilkan inovasi atau sekedar perbaikan sistem saja.

Dari enam (6) tahapan di atas, empat (4) tahapan akan dilaksakan di dalam penulisan skripsi ini, yaitu *initiation*, *planning*, *analysis*, dan *design*.

2.2.3 *Quality Engineering Memo (*QE Memo)

Memo adalah singkatan dari kata memorandum dapat bermakna macam-macam (1) memorandum berarti nota atau surat peringatan tidak resmi, (2) memorandum berarti surat pernyataan dalam hubungan diplomasi, dan (3) memorandum berarti bentuk komunikasi yang berarti saran, arahan, atau penerangan (Rohem, Fitria Nur : 2014). Memo merupakan bentuk komunikasi intra organisasi yang paling umum. Sebuah memo bisa jadi hanya berisikan satu atau dua paragraph saja dan dapat ditujukan untuk beberapa orang. Memo mempunyai beberapa tujuan seperti untuk menyebarkan kebijakan baru, melaporkan aktivitas tertentu, member instruksi, mengingatkan akan hal tertentu dll. Memo biasa digunakan untuk menyampaikan ide bisnis, rancangan atau membahas issue.

Memo juga merupakan bentuk alat untuk memberikan analis dengan berbagai konsep untuk menggambarkan setiap tampilan. Konsep-konsep disajikan pada berbagai tingkat detail dan presisi: kerangka kerja konseptual, pedoman untuk mengembangkan skenario (terutama untuk menggambarkan tugas / proses), heuristik yang membantu mengidentifikasi aspek-aspek penting, kuesioner terstruktur untuk memandu wawancara, dan template untuk mengumpulkan aspek formal.

Memo dalam *Quality Engineering* (QE Memo) adalah dokumen yang terstandard dan mempunyai format baku untuk memberikan informasi pada *department* lain ataupun *supplier* dengan tujuan mengurangi atau menyelesaikan *defect* yang ada di *production line* dan menjaga kualitas dari produk yang dikeluarkan. QE Memo adalah dokumen yang masuk dalam proses monitoring. Monitoring adalah proses pengumpulan dan analisis informasi berdasarkan indikator yang ditetapkan secara sistematis dan kontinu tentang kegiatan program sehingga dapat

dilakukan tindakan koreksi untuk penyempurnaan program kegiatan itu selanjutnya. Pemantauan yang dapat dijelaskan sebagai kesadaran (Awareness) tentang apa yang ingin diketahui, pemantauan berkadar tingkat tinggi dilakukan agar dapat membuat pengukuran melalui waktu yang menunjukan pergerakan ke arah tujuan atau menjauh dari itu (Hendini, Ade:2016). Dalam Memo ini terdapat detail mengenai *content* yang diajukan dan *background* dikeluarkannya memo tersebut. Memo ini diketahui oleh SPV dan disetujui hingga *Department Head*. Ada dua macam memo di *Quality Engineering* yaitu Memo yang mempunyai jangka waktu tertentu dan Memo yang jangka waktunya selamanya (*continues*). Karena QE Memo merupakan salah satu dokumen yang teraudit oleh tim audit maka QE Memo mempunyai nomer registrasi.

2.2.4 *User Interface*

Menurut Satzinger, *User Interface* adalah *input* dan *output* yang secara langsung melibatkan *user* sistem. *User Interface* dapat digunakan oleh pihak internal dan eksternal. Desain *user Interface* bisa bervariasi tergantung pada faktor-faktor seperti tujuan penggunaan *interface*, karakteristik pengguna, dan karateristik dari *device* untuk menampilkan *interface* (Satzinger, 2012).

Sebagai contoh, meskipun semua antarmuka pengguna harus dirancang untuk kemudahan maksimal penggunaan, ada beberapa pertimbangan lainnya, seperti efisiensi operasional, yang mungkin penting bagi pengguna internal yang dapat dilatih untuk menggunakan *interface* tertentu, dikombinasikan dan dioptimalkan untuk perangkat keras tertentu (misalnya, *keyboard*, *mouse*, dan layar resolusi tinggi besar).

2.2.5 Kerangka Pikir

Gambar 2. 12 Struktur Kerangka Berpikir

Dalam kerangka berpikir terdapat beberapa langkah, yaitu :

1. Menentukan topik penulisan yaitu strategi sistem informasi. Setelah itu menentukan objek penulisan yaitu QE Memo.
2. Mengidentifikasi masalah pada sistem yang sedang berjalan dengan cara melakukan penelitian secara langsung mengamati proses bisnis perusahaan tersebut tapi hanya di bagian QE Memo nya saja.
3. Menentukan tujuan, mafaat dan ruang lingkup agar tepat sasaran dan efektif.
4. Melakukan survey dan wawancara dengan *user* sebagai teknik pengumpulan data yang kami rasa efektif dan efisien untuk mendapatkan informasi.
5. Melakukan analisa dari data-data yang sudah kami kumpulan dengan metode *waterfall* hanya sampai langkah perancangan.
6. Mempelajari kondisi terkini proses bisnis perusahaan yang bersangkutan dengan QE Memo, lalu melakukan analisa sesuai dengan kondisi tersebut.

7. Setelah semua data dari perusahaan dan teori sudah lengkap baru kami melakukan perancangan untuk menjawab masalah pengguna.